Squares in a square: An On-line question

Andris Ambainis[1] and William Gasarch[2]
[1]Dept. of Comp. Sci., Univ. of CA at Berkeley, Berkeley, CA 94720, ambainis@cs.berkeley.edu
[2] Dept. of Comp. Sci., Univ. of MD, College Park, MD 20742, gasarch@cs.umd.edu

1. Introduction.

Soifer [5] conjectured that one can always place any finite collection of squares with area 1 inside a square of area 2 (with no overlapping). Stong [6] proved this conjecture. Stong's solution begins by sorting the squares by area.

We consider an *on-line* version of the problem. Let S be a square. We are given squares $sq_1, sq_2, \ldots, sq_n$ one at a time such that $\sum_{i=1}^{n} AREA(sq_i) \leq 1$. As soon as you see sq_i place it in S, so that you never have two squares overlapping. How big does S need to be in order to always be able to do this? We know that S must be have area at least 2 (this is easily seen to be true in the usual version of the problem). We will later show that S of area 4 suffices.

1) How big does S have to be? As an intermediary problem, find upper and lower bounds.
2) If the number of squares, n, is known ahead of time, then

how big does S have to be?

We show that area 4 suffices. Partition the 2×2 square into four 1×1 boxes. Hence, at the beginning, there are 4 empty boxes that are $2^0 \times 2^0$. At all stages there will be (1) some number of empty boxes of sizes $2^k \times 2^k$ for a variety of k, and (2) some number of partially filled boxes that we will never consider using. Assume that $sq_1, \ldots, sq_{n-1}$ have been placed.

ALGORITHM TO PLACE SQUARE sq_n

1) Let k be such that the length of a side of sq_n is in $(\frac{1}{2^k}, \frac{1}{2^{k-1}}]$.

2) Find the maximal $k' \leq k-1$ such that there is a $\frac{1}{2^{k'}} \times \frac{1}{2^{k'}}$ empty box.

3) If $k' = k-1$ then place sq_n in a corner of this box. (This box is now partially filled and can never be used again.) If $k' < k-1$ then split this box into four empty $\frac{1}{2^{k'+1}} \times \frac{1}{2^{k'+1}}$ boxes and go to Step 2.

END OF ALGORITHM

We show that if the algorithm is unable to place sq_n then $\sum_{i=1}^{n} AREA(sq_i) > 1$. It is easy to see that after a square is placed (1) the number of empty boxes of any given size is at most 3, and (2) if a partially filled box has been filled with a square of area a then the box has empty space of area strictly less then $3a$.

Assume, by way of contradiction, that sq_n cannot be placed and that $\sum_{i=1}^{n} AREA(sq_i) \leq 1$. Let k be such that the length of a side of sq_n is in $(\frac{1}{2^k}, \frac{1}{2^{k-1}}]$. By the algorithm there are no empty boxes of side $\frac{1}{2^{k-1}}$ or bigger. Hence all the empty boxes are of side $\frac{1}{2^k}$ or smaller. Since there are at most three empty boxes of any size the total area of the empty boxes is $\leq 3\sum_{i=k}^{\infty}(\frac{1}{2^k})^2 = \frac{1}{4^{k-1}}$. Since the partially filled boxes are filled with $sq_1, \ldots, sq_{n-1}$ they have empty space $< 3\sum_{i=1}^{n-1} AREA(sq_i) \leq 3(1 - AREA(sq_n)) \leq 3(1 - \frac{1}{4^k}) = 3 - \frac{3}{4^k}$ Hence the total amount of empty space is $< 3 - \frac{3}{4^k} + \frac{1}{4^{k-1}} = 3 + \frac{1}{4^k}$. So the total amount of filled space is greater than $4 - (3 + \frac{1}{4^k}) = 1 - \frac{1}{4^k}$. Since sq_n has area at least $\frac{1}{4^k}$ we have $\sum_{i=1}^{n} AREA(sq_i) > 1$. This is a contradiction.

2. Motivation

This problem is motivated by the fact that in computer science one often wants to study *on-line problems.* Memory allocation is typical. Suppose you have blocks of memory $B_1, B_2, \ldots$. We'll say each block has size N.

Offline Problem: Given a sequence of requests $r_1, r_2, \ldots, r_n$ for memory ($1 \leq r_i \leq N$), assign to each i a block B_j such that the sum of the requests assigned to any one block does not exceed N. Do this in a manner that minimizes the number of blocks used. This problem is NP-complete. The main obstacle to solving it is computational.

Online Problem: Given a sequence of requests $r_1, r_2, \ldots, r_n$ for memory, as soon as you get request r_i you must assign the request to a block. The goal is to minimize how many blocks are needed.

It is impossible to always achieve the optimal number of blocks [2],[7]. The main obstacle to solving it is informational. The best one can hope for is to have a solution that is within some constant times optimal. The best known result is that one can achieve 1.58872×OPTIMAL number of blocks [4].

For more on this particlar problem, usually called *bin packing*, see [3]. For more on online problems in general see [1].

References

[1] A. Borodin and R. El-Yaniv. *Online Computations and Competitive Analysis*. Cambridge Press, 1998.

[2] G. Galambos and J. Frenk. A simple proof of Liang's lower bound for on-line bin packing and extensions to the parametric case. *Discrete Applied Mathematics*, 41:173–178, 1993.

[3] E. Coffman, M.R. Garey, and D.S. Johnson. Approximation algorithms for bin packing: a survey. *Approximation algorithms for NP-hard problems*. Edited by D. Hochbaum. PWS publishing company, 46–93, 1997.

[4] M. Richey. Improved bounds for harmonic based bin-packing algorithms. *Discrete Applied Mathematics*, 34, 1991.

[5] A. Soifer. Squares in a square ii. *Geombinatorics*, 5(3):121,

1996.

[6] R. Stong. Squares inside of a square. *Geombinatorics*, 7(1):29–34, 1997.

[7] A. Yao. New algorithms for bin packing. *Journal of the ACM*, 27:207–227, 1980.